# PLATYPUS : A STEP-based Integration Framework

Alain Plantec, Vincent Ribaud

# PLATYPUS : A STEP-BASED INTEGRATION FRAMEWORK

A. Plantec,V. Ribaud
EA3883, LISyC, Université de Bretagne Occidentale, C.S. 93837 29238 Brest Cedex 3, France
E-mail: {Alain.Plantec, Vincent.Ribaud} @univ-brest.fr

***Abstract***

*STEP is an ISO standard (ISO-10303) for the computer-interpretable representation and exchange of product data. Parts of STEP standardize conceptual structures and usage of information in generic or specific domains. The standardization process of these constructs is an approach which can be applied to a data-based integration.*

*Platypus is a STEP-based meta-environment. From the data integration point of view, the STEP technology is primarily used for management. A key point is the automatic generation of a SDAI (a functional interface for STEP-modelled database independently of any particular system and language). Then, at the end of the complex process of defining standardized conceptual structures for the applications to be integrated, the STEP framework will provide a seamless access to data of different applications. Moreover, Platypus can be used to the development of a specific meta-environment intended to solve specific problems related to the legacy and new systems to be integrated.*

## 1 Introduction

D. S. Linthicum states that the need for a method of integrating disparate applications in a unified set of processes has emerged as a priority [1]. For a long time, a key issue in computer-aided software engineering (CASE) environments has been the desire to link tools that address different aspects of the development process. A. I. Wasserman categorizes five types of tool integration issues that must be addressed. These can be termed platform integration, presentation integration, data integration, control integration, and process integration [2]. Lessons learned from tools integration should be useful for application integration.

Enterprise Data Integration aims to share data and processing power from very different computer systems. The major benefits of integration are achieved when it is used to support a well-defined business process, but practically it is easier to share a consensus on business data and their usages rather than business processes. Hence, in this paper we address the problem of application integration from the data integration point of view, relying on our experience in CASE environments. We argue that application integration could be issued through a data-based integration process.

STEP is an ISO standard (ISO-10303) for the computer-interpretable representation and exchange of product data [3]. Parts of STEP standardize conceptual structures and usage of information in generic or specific domains. Standardized parts are defined with the EXPRESS language. The STEP design framework and the standardization process are presented in sections 2 and 3.

Platypus is a meta-case tool which embeds within a Squeak system a modelling and meta-modelling environment based on the STEP standard. Models and meta-models are specified with the EXPRESS language. The tools set of Platypus provides a strong support to the standardization process. Platypus together with the help provided to data integration are depicted in section 4.

# 2 The STEP standard

## 2.1 Overview

ISO 10303 provides a neutral mechanism for describing product data throughout the life cycle of a product, independent of any particular computer-aided system. ISO 10303 is suitable for file exchange and for implementing, sharing, and archiving product databases. The development of ISO 10303 is based upon the use of information models, a framework for product data modelling, formal data specification languages, and an architecture that separates information requirements from implementation methods.

**STEP description and implementation methods**

**The EXPRESS language** [1] is an object-oriented modelling language. The application data are described in schemata. A schema has the type definitions and the object descriptions of the application called Entities. An entity is made up of attributes and constraint descriptions.

**The STEP physical file format** defines an exchange structure using a clear text encoding of product data [2], for which a conceptual model is specified in the EXPRESS.

**The Standard Data Access Interface** (SDAI) [3] defines an access protocol for EXPRESS-modelled databases and is defined independently from any particular system and language.

The five main goals of the SDAI are: (1) to access and manipulate data which are described using the EXPRESS language, (2) to allow access to multiple data repositories by a single application at the same time, (3) to allow commit and rollback on a set of SDAI operations, (4) to allow access to the EXPRESS definition of all data elements that can be manipulated by an application process, and (5) to allow the validation of the constraints defined in EXPRESS.

**References**

[1] ISO 10303-11. Part 11 : EXPRESS Language Reference Manual, 1994.
[2] ISO 10303-21. Part 21 : Clear Text Encoding of the Exchange Structure, 1994.
[3] ISO 10303-22. Part 22 : Standard Data Access Interface, 1998.

## 2.2 The Standard Data Access Interface

The SDAI defines an interface between an application and the environment in which entity instances exist.

From the physical point of view, entity instances are stored into repositories. A repository is a data storage facility and may be implemented in memory, as a single database, multiple databases, a single file, a collection of files, or any other method. Within a repository, a distinction should be made between the different collections of instances (technically called SDAI-models) and schema instances. A SDAI-model is a logical container within which related entity instances exist. A schema instance provides the relationship between an EXPRESS schema and SDAI-models, and then defines the domain over which references between entity instances and rule validation are supported. A schema instance allows applications to have access to the information about the schema defining their data (data dictionary).

The SDAI operations are divided into several categories:

- system-level operations that allow an application to manage the transactions, repositories, …

- data dictionary-level operations that allow an application to manage the association of SDAI-models with schema instances and validate EXPRESS rules and references,
- data-level operations that allow an application to create, manipulate, delete and validate instances.

### 2.3 An example of a STEP use

In the three-dimensional Cartesian coordinate system, a point P in the xyz-space is represented by a tuple of three components (x,y,z). Any application working with Cartesian points should store points as tuples of three numbers in ASCII, binary or dedicated format. When n applications need a point-to-point communication, n*n pieces of software (format adapters) are required. STEP promotes the use of a standardized interface, able to read/write data in a neutral format. Thus, each application only needs an import and an export component, reducing to 2*n the number of required adapters.

A STEP environment, such as Platypus, provides a standardized management of schemata and instances. Moreover, the implementation of the SDAI in one or several programming languages is a major component of the environment. A re-engineering of an existing application for enterprise integration leads to develop the import/export adapters (depicted in grey on figure 1).

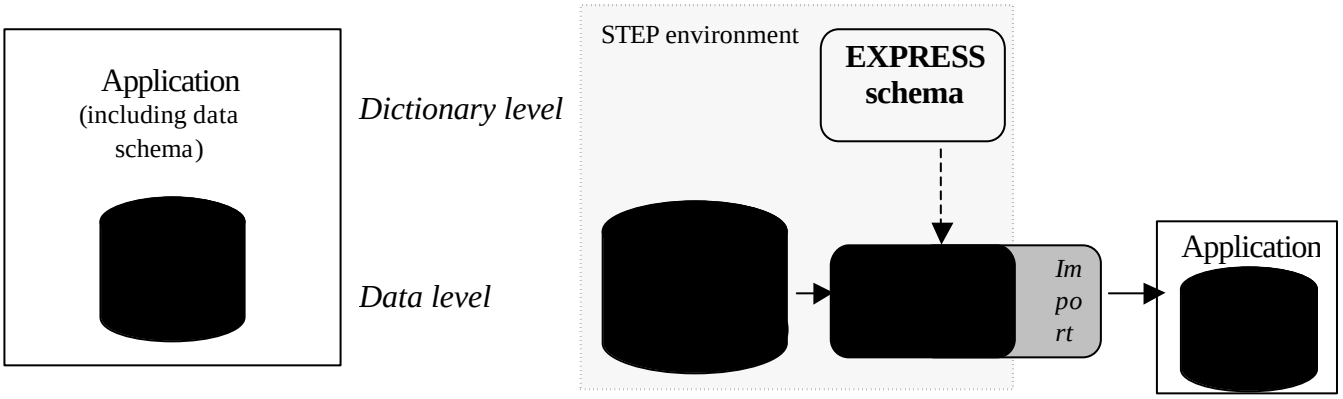


**Figure 1: Application architecture without or with STEP**

Without standardized interface, an application holds data in its own data structure. An application using STEP describes the structure of its data in an EXPRESS schema. From this schema, the STEP environment generates a SDAI in the application programming language. The SDAI provides a strong API which helps to develop import (respectively export) component able to read (respectively write) neutral data and load in (respectively extract from) the application.

```
SCHEMA Cartesian_coordinate_systems;
  ENTITY 3D_Cartesian_Point;
     x : REAL;
     y : REAL;
     z : REAL;
     derive
        ? : REAL:= sqrt(x**2 + y**2 + z**2);
  END_ENTITY;
END_SCHEMA ;
```

**Figure 2: Schema for the Cartesian coordinate system**

```
ISO-10303-21;
HEADER;
FILE_NAME('coordinate_systems.step', '25April 2006 4:58:11pm');
FILE_DESCRIPTION((''),'');
FILE_SCHEMA(('coordinate_systems'));
ENDSEC;
DATA;
#1=3D_CARTESIAN_POINT (0.1, 0.1, 0.1);
#2=3D_CARTESIAN_POINT 3D (0.2, 0.2, 0.2);
#3=3D_CARTESIAN_POINT 3D (0.3, 0.3, 0.3);
ENDSEC;
END-ISO-10303-21;
```

**Figure 3: Exchange file for the Cartesian coordinate system**

Another application could be reengineered in the same way. If both applications are using the same domain entities, applications can share the same EXPRESS schema which defines common entities, relationships and constraints. Each application is able to import/export a STEP file of instances and agrees on the same semantic of data. Thus, data integration is accomplished through schema sharing and neutral file exchange, as depicted on figure 4.

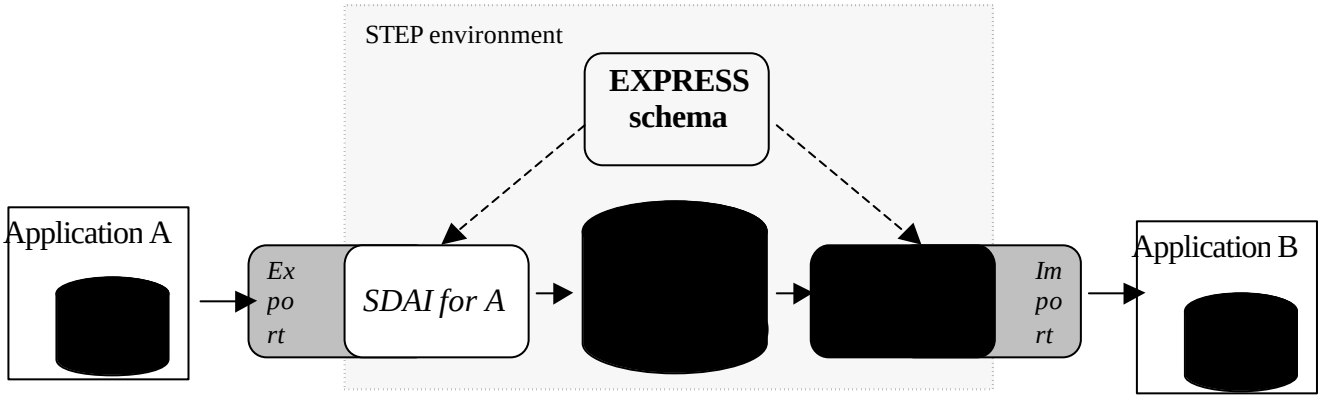


**Figure 4: Integration of applications through schema sharing and neutral file exchange**

# 3 The STEP integration framework

## 3.1 Role of application protocol

This section is a digest of excerpts from [4].

A fundamental concept of STEP is the definition of application protocol (APs) as the mechanism to specify information requirements and ensure reliable communication. An application protocol defines the context, scope, and information requirements for designated application(s) and specifies the STEP resource constructs used to satisfy these requirements.

The STEP integration framework establishes an explicit architecture for the conceptual models that are part of ISO 10303. This architecture provides the structure for the integrated resources and application protocols. The integrated resources provide constructs that are independent of a specific product data application context.

Application protocols employ three types of models.: an application activity model, an application reference model, and an application interpreted model.

**Application activity model (AAM):** a model that describes the activities and processes that use and produce product data in a specific application context. The AAM shall be defined in IDEF0, a formal process modelling language [5].

**Application reference model (ARM):** a model that specifies conceptual structures and constraints used to describe the information requirements of an application. The ARM shall be documented in formalized modelling language such as EXPRESS. Each information requirement has a normative definition.

**Application interpreted model (AIM):** a model of selected integrated resources which are constrained, specialized or completed to satisfy the information requirements of the ARM. The AIM shall be defined in EXPRESS.

### 3.2 A conceptual integration framework

An application is related to a domain (such as manufacturing, avionics). Each domain may dispose of one or several application protocols which define the form and the contents of the data used by the business activities of the domain.

The goal of an AP development process is the definition of an AIM. The AIM specification is based on the reuse of generic resources: an AIM is an EXPRESS schema that selects the applicable constructs from the integrated resources as baseline conceptual elements. Then, this schema is specialized with additional constraints, relationships and entities inheriting from generic constructs.

Integrated resources constructs are the conceptual structures and constraints widely accepted in any domain. These integrated resources constructs are interpreted in each application AIM by specifying global rules, for example, to modify optional attributes of entities to be non-existent or mandatory or to constrain entity behaviour; and by specifying subtypes of an entity to specialize the meaning of an attribute, localize a constraint on an entity and so on.

## 4 Working with the Platypus environment

### 4.1 Features

Platypus [http://cassoulet.univ-brest.fr:8000/Platypus] is a meta-case tool which embeds within a Squeak system, a modelling and meta-modelling environment based on the STEP standard. Let us depict some features.

**A STEP environment.** First of all, it is a STEP environment, allowing precise data models specification with STEP/EXPRESS modelling language and implementation of exchange components for models data which are described by EXPRESS models. From this point of view, Platypus is a typical STEP based tools with an EXPRESS editor and checker, and a STEP file reader, writer and checker.

**A hybrid data and object oriented development tool.** Platypus is implemented inside Squeak, a free Smalltalk environment. Thanks to Squeak, Platypus is an hybrid tool. On one hand, it allows very precise data specification and manipulation of strong typed objects. On the other hand, associated with Squeak code generator, it allows rapid system prototyping and efficient code maintenance.

**A data schema mapping environment.** Platypus is developed to be a schema mapping tool allowing the specification of mapping rules between source and destination schema.

Now, mapping rules are designed with EXPRESS and can be interpreted or used by a code generator.

### 4.2 Usages

We work with Platypus at two levels : application-level and case-tool level.

At the application-level, building an application follows three main steps:

- ?? data models are designed, including data types and validity rules definitions; these data models are referenced as domain models;
- ?? components related to domain models are generated in target programming language; these components not only include classes with data accessors but also all explicit rules and derived attributes (computed values) implementation;
- ?? generated classes are then enriched with application specific functionalities.

At the case-tool level, data handled are meta-data. Handled models are meta-models and the main application is code generation. We design and implement our code generators with Platypus, following a method that we have named Eugene (see, for example [6] about Eugene).

### 4.3 Back to the application integration

Enterprise Application Integration (EAI) is defined in [7] as the use of software and computer systems architectural principles to bring together (integrate) a set of enterprise computer applications. It is pointed out that EAI is not just about sharing data between applications; it focuses on sharing both business data and business process.

Our approach is restricted to the data. Designing the data schema is a kind of model integration. While data semantic is the sound of this integration, we should not forget that each application protocol relies on an application activity model (a functional point of view related to the domain) and that the SDAI defines a set of data management operations. Hence, there are a lot of functional services expressed in the data model by the underlying services and operations expected.

The SDAI is a functional interface for EXPRESS-modelled database and is independent of any particular system and language. The SDAI allows data sharing (through a common repository) as well as data exchange (through the use of a neutral format). The key point is that a SDAI is automatically generated from the EXPRESS schema of the application database (as long as a SDAI generator has been made for the target database management system). Then, at the end of the complex process of defining an Application Protocol for the applications to be integrated, the STEP framework will provide a seamless access to data of different applications. This ability is a first but fundamental step through the way of EAI.

### 4.4 A practical integration framework

Establishing an Application Protocol within a domain requires long-term efforts. Ultimately, the AP provides the framework for data integration because it provides a consensus on data definitions. But if this AP does not exist, we stated in section 2 that a common schema should provide a strong support to data integration.

A very common situation arises when two applications A and B have been developed independently from each other. We have to deal with a variety of different overlapping models, and

a major challenge is to provide a way to describe the semantical relationships of different models and to support the transformation of instances from one model to another one.

Integrating applications as depicted on figure 4 requires efforts at two different levels :

- modelling : the goal is to define a schema which contains the concepts common to both underlying models;
- programming : the challenge is to be able to transform data from an application into the neutral representation; even with the help of the SDAI, this task is much heavier than modelling.

It could be envisaged to work with two EXPRESS schemata, one close to the application A model, the other close to the application B model. Thus, programming the transformation of A (resp. B) data into a neutral representation is made easier because the EXPRESS schema for A (resp. B) is as close as possible from the own application data structure. The main problem is to migrate instances from a neutral representation to another.

Fortunately, EXPRESS provides some support to schema integration and instances migration. Any schema can use constructs from another schema (with the USE of REFERENCE clauses) and derived attributes can be specified with procedural and queries features.

### 4.4.1 An example of derivation

In the spherical coordinate system, a point P is represented by a tuple of three components (?,?,F). Using terms of the Cartesian coordinate system :

- ?? 0 = ? (radius) is the distance between the point P and the origin,
- ?? 0 = F = 180 (zenith, colatitude or polar angle) is the angle between the z-axis and the line from the origin to the point P, and
- ?? 0 = ? = 360 (azimuth or longitude) is the angle between the positive x-axis and the line from the origin to the point P projected onto the xy-plane.

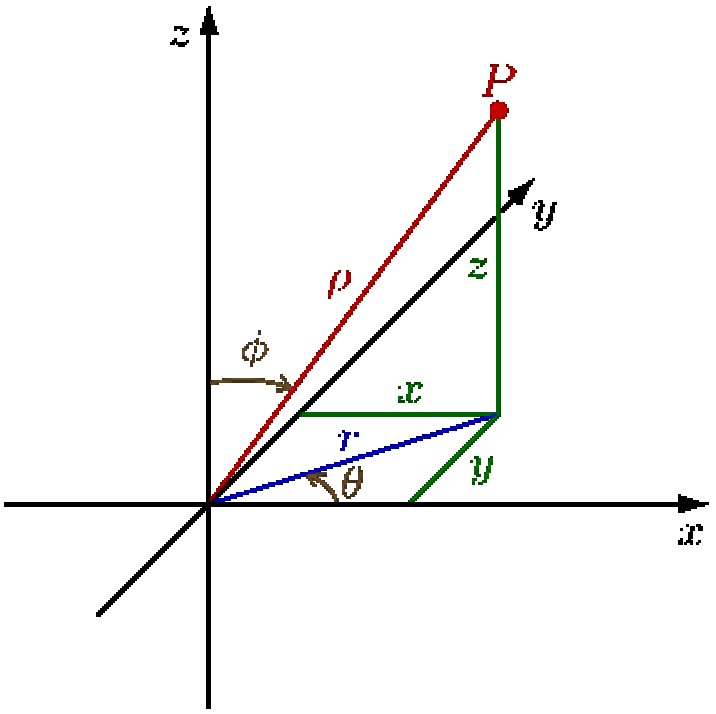


**Figure 5: Cartesian and spherical coordinates (from Wikipedia [8])**

We can define a schema for the spherical coordinate systems and express its relationship with the Cartesian coordinate.

```
SCHEMA Spherical_coordinate_systems;

  ENTITY Spherical_Point;
    ? : REAL;
    ? : REAL;
    F : REAL;
  END_ENTITY;
END_SCHEMA ;

SCHEMA Cartesian_coordinate_systems;
USE Spherical_coordinate_systems;
  ENTITY 3D_Cartesian_Point;
     x : REAL := ? * cos (F) * cos (?);
     y : REAL := ? * cos (F) * sin (?);
     z : REAL := ? * sin (F);
  END_ENTITY;
END_SCHEMA ;
```

**Figure 6: An example of schema for coordinate systems**

### 4.4.2 Impact on application integration

When a schema B uses (or references) another schema A, A constructs are available within B hence the SDAI for B is able to read/write instances conforming to A schema. If the derivation mechanism is used to indicate how B constructs rely on A constructs, the STEP environment is able to load a neutral file of A instances and to compute the instanciation of corresponding B instances. Hence the transformation between A and B representations is automatic and provided by the SDAI generated from A and B schemata.

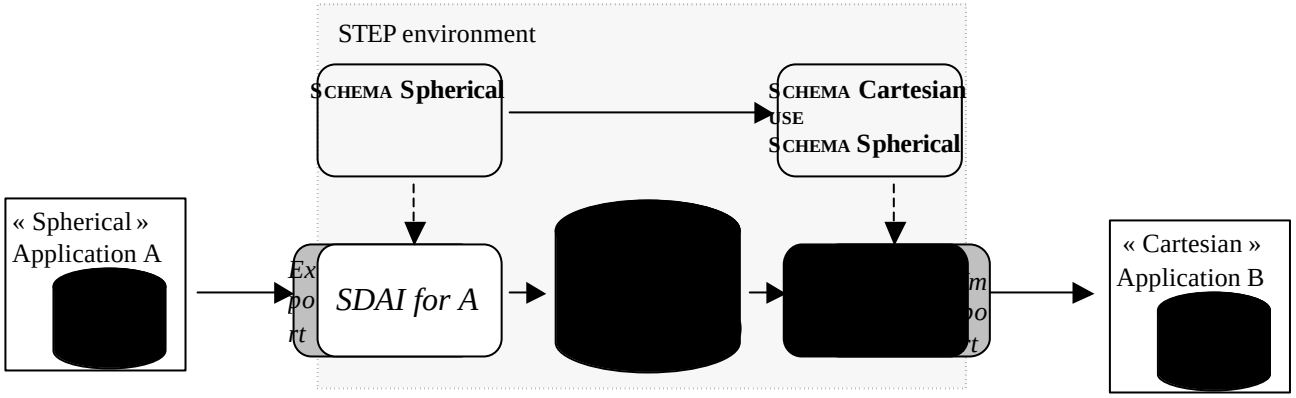


**Figure 7: Integration of applications through schema transformation**

The figure 7 shows the process of integration applied to an application A using spherical coordinates and an application B using Cartesian coordinates. Little effort has to be made to develop import/export adapters because :

- the EXPRESS schema for each application is very close to the data structure used;
- the generated SDAI provides a strong support to handle the application data.

### 4.5 Advanced features

As pointed out in [7], if integration is applied without following a structured EAI approach, many point-to-point connections grow up across an organization. It could often happen that an ad-hoc solution to integrate data between two applications should be applied to the whole set of applications (for example, transforming indexed files into relational tables). The generic solution no more lies in dealing with a particular indexed file to be transformed into a particular table, but in the concept of indexed files and the concept of relational tables.

Working at a generic level often means that we are working with meta-models rather than with models. The process works on a source and a target meta-model which are the description of the source (i.e. indexed files) and the target (i.e. relational tables) language constructs. The generic solution is described through generation rules specifying how to translate source constructs into target constructs.

The translation between the reference model and application models is not straight forward and we need a model transformation environment. All the models are specified with EXPRESS and the corresponding meta-model is the EXPRESS meta-model (defined itself in EXPRESS). In a meta-model, the entities specify the definitions of the concepts of the domain. The constraints specify their semantics locally or globally. As entities and attributes mapping rules can be specified by derived attributes, Platypus can be used as a schema mapping tool and as a code generator builder.

Meta-models used are specialization of the EXPRESS meta-model. Thus, tools of the Platypus framework (taxonomy browser, structured editor, analyser, code generator ...) can be reused and specialized.

## 5 Conclusion

STEP is an ISO standard (ISO-10303) for the computer-interpretable representation and exchange of product data. A fundamental concept of STEP is the definition of application protocol (APs) as the mechanism for specifying information requirements and ensuring reliable communication. The standardization process of these APs is an approach which can be applied to a data-based integration.

Platypus is a STEP-based meta-environment. The STEP technology is used for meta-models and models management. As a STEP-based environment, Platypus provides a support to data exchange and data sharing. Moreover, Platypus can be used for the development of a specific meta-environment intended to solve specific problems related to the legacy and new systems to be integrated.

## 6 References


[1] David S. Linthicum, Enterprise Application Integration, Addison-Wesley Information Technology Series, 1999.

[2] Anthony I. Wasserman, Tools Integration in Software Environments, in International Workshop on Environments, LNCS 467, Springer-Verlag , 1990.

[3] ISO 10303-1. Part 1 : STEP overview and fundamentals principles, ISO, 1994.

[4] ISO TC184/SC4 N433 Guidelines for the development and approval of STEP application protocols, version 1.2, 1996.

[5] Federal Information Processing Standard Publication 183, Integration Definition for Function Modeling (IDEF0), FIPS PUB 183, National Institute of Standards and Technology, December 1993.

[6] Alain Plantec and Vincent Ribaud. EUGENE: a STEP-based framework to build Application Generators. AWCSET'98, CSIRO-Macquarie University, 1998.

[7] Wikipedia Enciclopedia: http://en.wikipedia.org/wiki/Enterprise_Application_Integration, 2005.

[8] Wikipedia Enciclopedia: http://en.wikipedia.org/wiki/Spherical_coordinates, 2006.